第二十二节 稳定性升级

Dubbo URL 设计

Dubbo URL 和标准 URL 是有一些差别，标准 URL 更为严格

Dubbo URL 是 Dubbo 贯穿整个框架的元信息载体

```
consumer://
provider://
dubbo://
```

Dubbo URL 变为 Dubbo 执行中的上下文

消费服务 URL 通常服务提供方的主机 IP 和端口是不同，但是服务名称一样，比如 com.acme.UserService

https://mercyblitz.github.io/2020/05/11/Apache-Dubbo-%E6%9C%8D%E5%8A%A1%E8%87%AA%E7%9C%81%E6%9E%B6%E6%9E%84%E8%AE%BE%E8%AE%A1/

Dubbo SPI 扩展实现

扩展点加载

SPI 注解 - @SPI

SPI 扩展加载器 - org.apache.dubbo.common.extension.ExtensionLoader

- 兼容 JDK SPI ServiceLoader - `META-INF/services`（不推荐使用）
- `META-INF/dubbo/` - Dubbo 外部应用扩展
- `META-INF/dubbo/internal/` - Dubbo 内部框架实现
- `META-INF/dubbo/external/` - Dubbo 外部框架实现

场景： SPI 有多种实现来源，第一种是 Dubbo 内部，第二种是应用扩展（Dubbo 外部）

假设 Dubbo 内部能够覆盖外部的话，那么 Dubbo 内部负载均衡永远无法同名的。

SPI 加载策略 - org.apache.dubbo.common.extension.LoadingStrategy

1. org.apache.dubbo.common.extension.DubboInternalLoadingStrategy
    - META-INF/dubbo/internal/
    - abc = com.acme.OldAbc
    - overridden = false
2. org.apache.dubbo.common.extension.DubboExternalLoadingStrategy
    - META-INF/dubbo/external/
    - abc = com.acme.Abc
    - overridden = true
3. org.apache.dubbo.common.extension.DubboLoadingStrategy
    - META-INF/dubbo/
    - abc = com.acme.NewAbc
4. org.apache.dubbo.common.extension.ServicesLoadingStrategy
    - META-INF/services

abc

- org.apache.dubbo.common.extension.DubboInternalLoadingStrategy
    - abc = OldAbc

- org.apache.dubbo.common.extension.DubboExternalLoadingStrategy（overridden = true）
  - abc replace OldAbc -> Abc
- org.apache.dubbo.common.extension.DubboLoadingStrategy（overridden = true）
  - abc replace Abc-> NewAbc

NewAbc -> Abc -> OldAbc

Dubbo Spring 整合实现

Dubbo Config 模块

基类 - org.apache.dubbo.config.AbstractConfig

- 均有一个 id

存储 - org.apache.dubbo.config.context.ConfigManager

配置类	标签	用途	解释
`ProtocolConfig`	`<dubbo:protocol/>`	协议配置	用于配置提供服务的协议信息，协议由提供方指定，消费方被动接受
`ApplicationConfig`	`<dubbo:application/>`	应用配置	用于配置当前应用信息，不管该应用是提供者还是消费者
`ModuleConfig`	`<dubbo:module/>`	模块配置	用于配置当前模块信息，可选
`RegistryConfig`	`<dubbo:registry/>`	注册中心配置	用于配置连接注册中心相关信息
`MonitorConfig`	`<dubbo:monitor/>`	监控中心配置	用于配置连接监控中心相关信息，可选

配置类	标签	用途	解释
`ProviderConfig`	`<dubbo:provider/>`	提供方配置	当 ProtocolConfig 和 ServiceConfig 某属性没有配置时，采用此缺省值，可选
`ConsumerConfig`	`<dubbo:consumer/>`	消费方配置	当 ReferenceConfig 某属性没有配置时，采用此缺省值，可选
`MethodConfig`	`<dubbo:method/>`	方法配置	用于 ServiceConfig 和 ReferenceConfig 指定方法级的配置信息
`ArgumentConfig`	`<dubbo:argument/>`	参数配置	用于指定方法参数配置

Dubbo 编程模型

API 编程模型

服务提供者 - ServiceConfig

服务消费者 - ReferenceConfig

Spring XML 配置编程模型

XML 结构 - XML Schema

Spring XML Schema 资源 与物理路径映射 - META-INF/spring.schemas

Spring XML 元素处理器 - META-INF/spring.handlers

服务提供者 - `<dubbo:service>`

ServiceConfig 中的 Properties 均被 `<dubbo:service>` 属性（Attributes）来设置

服务消费者 - `<dubbo:reference>`

Spring 注解驱动编程模型

服务提供者 - @DubboService

@DubboService 所标注的 Class 注册为普通 Spring Bean（类似于 @Service，应用服务），同时，将 @DubboService 转化为 ServiceBean（Spring Bean，Dubbo 组件），比如：

```
package com.acme; //
@EnableDubbo(scanBasePackages="com.acme") +
@EnableDubbo(scanBasePackages="com")

@DubboService
@Service // Spring @Service 注解
public class DefaultDemoService implements
DemoService { // 类似于 @Service 注册方法
}

//
ServiceBean<DemoService> // Spring Bean，何时暴露
DemoService 服务？
// Bean 生命周期，还是 IoC 容器生命周期
// 理想时机：应用启动完毕，服务消费端消费前
```

简言之，一个 @DubboService Class 将注册两个 Spring BeanDefinition：

- 目标服务 BeanDefinition
- ServiceBean BeanDefinition
    - 注入一个目标服务 BeanDefinition

服务提供者 - @DubboReference

近端调用

@DubboService A (a)

@DubboReference A (a)

org.apache.dubbo.config.spring.context.DubboBootstrapApplicationListener

https://mercyblitz.github.io/2018/01/18/Dubbo-%E5%A4%96%E9%83%A8%E5%8C%96%E9%85%8D%E7%BD%AE/

Dubbo 核心组件实现

服务暴露和引用（消费）

实现细节 - https://dubbo.apache.org/zh/docs/v2.7/dev/implementation/

注册中心

核心 API - org.apache.dubbo.registry.Registry

容错能力的抽象实现 - org.apache.dubbo.registry.support.FailbackRegistry

- 注册方法 - org.apache.dubbo.registry.support.FailbackRegistry#register
- 子类实现 - org.apache.dubbo.registry.support.FailbackRegistry#doRegister
- 服务订阅 - org.apache.dubbo.registry.zookeeper.ZookeeperRegistry#doSubscribe
  - 同步定义
  - 异步更新

服务目录监听器 - org.apache.dubbo.registry.NotifyListener

Zookeeper 实现注册中心

Zookeeper 存储结构是树形（目录），默认根节点：
dubbo，可以通过 group URL 参数调整

路径

ServiceInterface + / + providers + /

/dubbo/com.acme.XXXService/providers/
${URL.toFullString()}

- /dubbo/com.acme.XXXService/providers/${dubbo://127.0.0.1:20880/com.acme.XXXService?version=1.0.0}
- /dubbo/com.acme.XXXService/providers/${rest://127.0.0.1:20881/com.acme.XXXService?version=1.0.0}
- /dubbo/com.acme.XXXService/providers/${thift://127.0.0.1:20882/com.acme.XXXService?version=1.0.0}
- /dubbo/com.acme.XXXService/providers/${dubbo://127.0.0.2:20880/com.acme.XXXService?version=1.0.0}
- /dubbo/com.acme.XXXService/providers/${rest://127.0.0.2:20881/com.acme.XXXService?version=1.0.0}
- /dubbo/com.acme.XXXService/providers/${thift://127.0.0.2:20882/com.acme.XXXService?version=1.0.0} (new)

Nacos 实现注册中心

DataId

${protocol}:${service-interface}:${version}:${group}

dubbo:ServiceInterface:version:group

GroupId

DEFAULT_GROUP

dubbo:ServiceInterface:version:group

ServiceInstance

127.0.0.1:20880

127.0.0.2:20880

自定义实现注册中心

实现 SPI - org.apache.dubbo.registry.RegistryFactory

实现 org.apache.dubbo.registry.Registry

基本流程

ReferenceConfig -> 注册中心中去订阅它关联的服务接口 -> 得到该接口暴露 URL 列表 -> 转化为 Invoker 集合 -> 通过 Router 实现路由出 Invoker （子）集合 -> 再经过 LoadBalance 选择其中一个 Invoker -> 执行远程调用 -> Exchange -> Transport -> Serialization

Dubbo 设计不足

1. DubboBootstrap 与 ServiceConfig 耦合

   DubboBootstrap 是一个 Dubbo 顶层 API，引导 Dubbo 应用启动，类似于 Netty 实现。

   ServiceConfig 引导单个服务上线和下线。

其他

SOA 时代和 MSA 时代

SOA 时代通常有行业解决方案，比如 Oracle、BEA、SUN ApplicationServer

MSA 时代强调自建，12 factors -> Cloud-Native

作业

通过 Java 实现两种（以及）更多的一致性 Hash 算法

（可选）实现服务节点动态更新

org.apache.dubbo.rpc.cluster.loadbalance.ConsistentHashLoadBalance